Using the IOT Toolkit for I/O and MPI Performance Analysis

IOT is a licensed toolkit developed by I/O Doctors, LLC, for I/O and MPI instrumentation and optimization of high-performance computing programs. It allows flexible and user-controllable analysis at various levels of detail: per job, per MPI rank, per file, and per function call. In addition to information such as time spent, number of calls, and number of bytes transferred, IOT also provides the time of the I/O and/or MPI call, and where in the file the I/O occurs.

IOT can be used to analyze Fortran, C, and C++ programs as well as script-based applications, such as R, MATLAB, and Python. The toolkit works with multiple MPI implementations, including HPE MPT and Intel MPI. It is available for use on Pleiades, Electra, and Endeavour. Basic instructions are provided below. If you are interested in more advanced analysis, contact User Services at support@nas.nasa.gov.

Setting Up IOT

To set up IOT, complete the following steps. You only need to do these steps once.

1. Add **`/nasa/IOT/latest/bin64`** to your search path in your .cshrc file (for csh users) or .profile file (for bash users), as shown below. Be sure to add this line *above* the line in the file that checks for the existence of the prompt.

   For csh, use:
   `set path = ( $path /nasa/IOT/latest/bin64 )`

   For bash, use:
   `PATH=$PATH:/nasa/IOT/latest/bin64`
2. Run the **`iot -V`** command to check whether IOT is working. The output should be similar to the following example:

   ```
   % iot -V
   Using IOT install /nasa/IOT/v4.0.04/
           bin64/iot            v4.0.04 built Jun 14 2017 11:23:19
           lib64/libiot.so      v4.0.04 built Jun 14 2017 11:23:19
           libiotperm.so  v3.2.08 "Nasa_Advanced_SuperComputing" 5/11/2018 *
   ```
3. In your home directory, untar the **`/nasa/IOT/latest/ipsd/user_ipsd.tgz`** file:

   ```
   % tar xvzf /nasa/IOT/latest/ipsd/user_ipsd.tgz
   ```

   A directory called **`ipsd`** should be created under your $HOME directory.
4. Confirm that **`ipsd`** can be started:

   ```
   % ipsctl -A `hostname -s`
   No shares detected
   ```
5. Test IOT using the **`dd`** utility. First, create a directory called "dd" and change (**`cd`**) into it. Then, run the **`iot`** command as follows:

   ```
   % iot dd if=/dev/zero of=/dev/null count=20 bs=4096
   20+0 records in
   20+0 records out
   81920 bytes (82 kB) copied, 0.000123497 s, 663 MB/s
   ```

   In addition to the output shown above, you should also find a file called **`iot.`*`xxxxx`*`.ilz`**. The ILZ file is the output from the **`iot`** command.

Using IOT to Analyze Your Application

Once IOT is set up, follow these steps to analyze your application.

1. Create a configuration file that tells IOT what you want to instrument or monitor. You can use one of the following sample configuration files, which are available in the **`/nasa/IOT/latest/icf`** directory:

   trc_summary.icf
   : Use this file to start your first I/O analysis. This file provides a summary of information on the total counts, time spent, and bytes transferred for each I/O function of each file. For MPI applications, it also provides the same information obtained with mpi_summary.icf (described below).

   trc_interval.icf
   : In addition to the data collected by trc_summary.icf, this file provides more details for the read/write MPI function, per 1000-ms interval, including: the wall time when the calls occur; counts; time spent; and bytes transferred.

   trc_events.icf
   : In addition to the data collected by trc_summary.icf, this file provides the most details for each read/write function at the per-event level, including: the wall time when each call occurs; the time spent for the call; and the number of bytes transferred.

   mpi_summary.icf
   : Use this file to start your first MPI analysis. The file provides a summary of information such as the total count, time spent, and bytes transferred for all of the MPI functions called by the MPI ranks.

   mpi_interval.icf
   : In addition to the information collected by mpi_summary.icf, this file provides more details for the MPI functions, per 1000-millisecond (ms) interval, including: the wall time when the calls occur; counts; time spent; and bytes transferred.

   mpi_events.icf
   : This file provides the most details for each MPI function, at the per-event level, including: the wall time when each call occurs; the time spent for the call; and the number of bytes transferred.
2. Modify the **`mpiexec`** execution line in your PBS script to run IOT. For example, replace **`mpiexec -np 100 a.out`** with the following lines:

   ```
   set JOB_NUMBER=`echo $PBS_JOBID | awk -F. '{ print $1 }'`
    iot -m mpt -f cfg.icf -c '$':pfe22:`pwd`/a.out.collect.$.ilz \
    mpiexec -np 100 a.out
   ```

   This method will generate an ILZ file named **`a.out.collect.$.ilz`**.

   Another option is to simply use:

   ```
   iot -m mpt -f cfg.icf \
    mpiexec -np 100 a.out
   ```

   This method will generate an ILZ file named **`iot.`*`process_id`*`.ilz`**.

   TIP: When the **`-m`** option is enabled, the default value for **`-f`** is **`mpi_summary.icf,`** which will be located automatically in the **`/nasa/IOT/latest/icf`** directory.

For more information, see **`iot -h`** on the **IOT Options** and **`iot -M`** on the **IOT Layers** man pages.

Viewing the ILZ File

Once your ILZ file is generated, you can view the data with the Pulse graphical user interface (GUI) using one of the following methods. Pulse will read in the data from the file and organize it for easy analysis in the GUI.

Run Pulse on Your Local System (Recommended)

Follow these steps to run Pulse on your local system.

1. Download Pulse.jar and the ILZ file:

   ```
   your_local_system% scp pfe:/nasa/IOT/latest/java/Pulse.jar .
   your_local_system% scp pfe:/path_to_ilz_file/filename.ilz .
   ```
2. Run Pulse through Java:

   ```
   your_local_system% java -jar Pulse.jar filename.ilz
   ```

Note: Download the latest version of Pulse.jar from time to time, as enhancements may be added.

Run Pulse from a PFE

Log into a PFE, load a Java module, and run Pulse:

```
pfe21% module load jvm/jrel.8.0_121
pfe21% pulse filename.ilz
```

TIPS:

- Pulse will uncompress the ILZ file to 4-5 times its compressed size. If the uncompressed file gets very large, Pulse may run out of memory. If this happens, you can try to increase memory using the **`java -Xmx4g`** option, as follows:

  ```
  % java -Xmx4g -jar Pulse.jar filename.ilz
  ```
- While Pulse is reading the file, the filename in the GUI will appear in red text. You can stop it before Pulse consumes too much memory by right-clicking the filename and selecting **Stop Reading**.

Additional Documentation

IOT documentation provided by the vendor is available in the **`/nasa/IOT/Doc`** directory.

Article ID: 546
Last updated: 13 May, 2021
Revision: 78
Optimizing/Troubleshooting -> Code Development -> Performance Analysis -> Using the IOT Toolkit for I/O and MPI Performance Analysis
https://www.nas.nasa.gov/hecc/support/kb/entry/546/